Towards an Optimized Illumina Microarray Data Analysis Pipeline

Pan Du, Simon Lin

Robert H. Lurie Comprehensive Cancer Center,
Northwestern University

August 06, 2007
Outline

• Introduction of Illumina Beadarray technology
• Lumi package and its unique features
  – VST (variance stabilizing transform)
  – RSN (robust spline normalization)
  – nuID annotation packages
• Analysis pipeline
• Example R code
Illumina BeadArray Technology

Two formats: 96-sample Array Matrix and the multi-sample BeadChip formats

Uniform pits are etched into the surface of each substrate to a depth of approximately 3 microns prior to assembly.

Each type of bead has about 30 technique replicates

Beads are then randomly assembled and held in these microwells by Van der Waals forces and hydrostatic interactions with the walls of the well.
## (Previous) Concerns

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Illumina Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Uneven distribution of BG (air bubble and washing)</td>
<td>• Larger number of beads</td>
</tr>
<tr>
<td>• Contamination of debris</td>
<td>• Random distribution of beads</td>
</tr>
<tr>
<td>• Scratches on the surface</td>
<td></td>
</tr>
<tr>
<td>Spot morphology and uniformity</td>
<td>Coated beads instead of printing</td>
</tr>
<tr>
<td>Array manufacturing defect</td>
<td>Tested in the decoding process</td>
</tr>
<tr>
<td>Failure in labeling of mRNA</td>
<td>Labeling control on array</td>
</tr>
<tr>
<td>Scanning conditions</td>
<td>¿ Still a concern ?</td>
</tr>
</tbody>
</table>
## (Previous) Concerns (continued)

<table>
<thead>
<tr>
<th>Challenges</th>
<th>Illumina Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Probe Specificity</td>
<td>50-mer design</td>
</tr>
<tr>
<td>Normalization issues</td>
<td>6 to 12 arrays on the same slide</td>
</tr>
</tbody>
</table>

### Illumina Solutions

- **Probe Specificity**: 50-mer design
- **Normalization issues**: 6 to 12 arrays on the same slide
• Slide: 2 cm x 7cm
• Bead: 3 um
Illumina Terminology (UML diagram)
Each array is different
Unique Features of Illumina Microarray

- Multiple samples on the same chip. Reduce the batch effects and cost per sample.
- Randomness
  - Randomly arranged beads, each of which carries copies of a gene-specific probe.
- Redundancy
  - For each type of beads, there are about 30 randomly positioned replicates.
- The preprocessing and quality control of Illumina microarray should be different from other types of microarrays.
Other Bioconductor Packages for Illumina Microarray

- *beadarray* is mainly designed for quality control and low-level analysis of BeadArrays.
- *BeadExplorer* is aimed to provide data exploration and quality control by leveraging the output of *Illumina BeadStudio* and existing algorithms in the *affy* package;
- Does not take the advantage of larger number of technical replicates available on the Illumina microarray in the preprocessing.
Design Objectives of *lumi* Package

- To provide algorithms uniquely designed for Illumina
- To best utilize the existing functionalities by following the class infrastructure and identifier management framework in Bioconductor
Functionalities of *lumi* Package

- Range from data import, quality control, preprocessing to gene annotation
- S4 class based and compatible with other Bioconductor packages
- Support the general microarray preprocessing algorithms
Unique features of *lumi* package

- Intelligent read data from BeadStudio output file
- A variance-stabilizing transformation (VST) algorithm
- A robust spline normalization (RSN) method
- The nucleotide universal identifier (nuID) identified annotation packages
Object Models

- Design based on the S4 Classes.
- One major class: lumiBatch
- Compatible with other Bioconductor packages;
Intelligently Load the Data

- Automatically recognize the format and different versions of the BeadStudio output
- Automatically check the identifiers (TargetID or ProbeID) used, and replaced them as nuID if the annotation library is provided.
- Extract the sample information from the sample IDs.
- `lumi.x <- lumiR(fileName)`
Variance Stabilization

• General assumption of statistical tests to microarray data: variance is independent of intensity
• In reality, larger intensities tend to have larger variations
• Current implementation:
  – Log2 transform is widely used
  – VSN (Variance Stabilizing Normalization) combines variance stabilizing and normalization based on the limited replicates across chips
Variance Stabilizing Transformation (VST)

- Taking the advantage of larger number of technical replicates available on the Illumina microarray
- A generalized log transformation (arcsinh function)
Variance Stabilization of the Bead Level Data

(A) Raw data

(B) VST transformed

(C) Log2 transformed

(D) Cubic root transformed
Variance Stabilization of the Technical Replicates
Robust Spline Normalization (RSN)

- Quantile normalization:
  - Pros: computational efficiency, preserves the rank order
  - Cons: The intensity transformation is discontinuous
- Loess and other curve-fitting based normalization:
  - Pros: continuous
  - Cons: cannot guarantee the rank order. Strong assumption (majority genes unexpressed and symmetric distributed)
- RSN combines the good features of the quantile and loess normalization
Comparison of curve fitting and quantile normalization

<table>
<thead>
<tr>
<th></th>
<th>Curve fitting based normalization</th>
<th>Quantile normalization</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Assumption</strong></td>
<td>Most genes are not differentially expressed.</td>
<td>All samples have the same distribution.</td>
</tr>
<tr>
<td><strong>Approximation</strong></td>
<td>Based on curve fitting</td>
<td>Replaced by the average of the probes with the same rank</td>
</tr>
<tr>
<td><strong>Problems</strong></td>
<td>Does not work well when lots of genes are differentially expressed.</td>
<td>Will lose small difference between samples, and the change is unrecoverable. Normalize across all samples, memory intensive.</td>
</tr>
<tr>
<td><strong>Strengths</strong></td>
<td>The value mapping is continuous. Normalize in pairwise, memory save.</td>
<td>Rank invariant Computationally efficient</td>
</tr>
</tbody>
</table>
Robust Spline Normalization (RSN)

- Combining the strength of curve fitting and quantile normalization
  - Continuous mapping
  - Rank invariant
  - Insensitive to differentially expressed genes.

- Basic Ideas of RSN
  - Perform a quantile normalization of the entire microarray dataset for the purpose of estimating the fold-changes between samples
  - Fit a weighted monotonic-constraint spline by Gaussian window to down-weight the probes with high fold-changes
  - Normalize each microarray against a reference microarray
Performance Evaluation
nuID Illumina annotation packages

- Avoiding the inconsistence of the identification system currently used by Illumina microarray
- nuID is one-to-one correspondence to the probe sequence
- nuID is self-identifiable and including error checking
- Get the most updated probe annotation by blasting the RefSeq database
Analysis Pipeline
Example Code

```r
# load the library
library(lumi)

# specify the file name output from Bead Studio
fileName <- 'Barnes_gene_profile.txt'
# Read the data and create a LumiBatch object
example.lumi <- lumiR(fileName, lib='lumiHumanV1')

## summary of data
example.lumi
## summary of quality control information
summary(example.lumi, QC )

## preprocessing and quality control after normalization
lumi.N.Q <- lumiExpresso(example.lumi)
## summary of quality control information after preprocessing
summary(lumi.N.Q, QC )

## plot different plots
pairs(lumi.N.Q)
plot(lumi.N.Q, what='sampleRelation')
boxplot(lumi.N.Q)

# Extract expression data for further processing
dataMatrix <- exprs(lumi.N)
```
Benchmark Data Sets to compare Methods

<table>
<thead>
<tr>
<th></th>
<th>library (affy)</th>
<th>library (lumi)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data sets</td>
<td>affycomp</td>
<td>lumiBarnes</td>
</tr>
<tr>
<td>Latin-square</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>spike-in?</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Titration?</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>
Part II: nu1D
A Novel Identifier for Oligos, Ideal for Oligonucleotide-based Microarrays
Outline

• What is nuID?
• Why nuID?
• How does nuID work?
What is nuID

- nuID is the abbreviation of Nucleotide Universal Identifier
- nuID is a novel identifier for oligos, ideal for oligonucleotide-based microarrays
Why nuID?
Microarray Information Flow

GO:0051301  Cell Division

entrezID: 19645  symbol: Rb1

retinoblastoma 1
1 ggcgggccc gtcgggtti cctcggggga gttcccatca tttttgaacc gggantcggg
61 tgagggaggg gttgcccccg gttgcgcgcg cgacccgccc cctccccggg ggcgtcctctc
121 ggtgtgtcgc ggcgggccc gcgtgcgcgtc atgcgcgcca aagccccggc cagagccggc
181 ggcgcgggac ccgcgceacc ggcgcggcgc cgcctcgggc aggacgccg ccgcccaggac
genome 241 agcgcccccg aagagctgcc cctgccccgg ctctggttgg aagaaattga agaaacccgaa
301 ttattgcct tatgtcctaa gttaaaggtc cccgatcatg tcagagaag agcttggtct...

ID: ??  ggtaccccgatcatgtcagagaa

8/10/07  Bioconductor 2007  32
observed

Probe 1
Probe 2
Probe 3
Probe 4
Probe 5
... 
Probe N

dynamic

stable

Gene 1
Gene 2
Gene 3
... 
Gene M

dynamic

inferred

GO 1
GO 2
... 
GO P

dynamic

Entity

Mapping

8/10/07 Bioconductor 2007 33
For Illumina microarrays, TargetID was used as the primary ID in the NCBI GEO database.
Challenges of Target IDs

- Not unique: “GI_28476905” and “scl0076846.1_142” are the same gene on Mouse_Ref-8_V1 chip.
  -- Synonyms.

- Not stable over time: “GI_21070949-S” in the Mouse_Ref-8_V1 chip but as “scl022190.1_154-S” in the later Mouse-6_V1 chip.
  -- IDs can be recycled or retired.

- Not universal across manufacturers
  -- Homonyms.

- Not interpretable without metadata: However, metadata (lookup table) is not always available in reality.
How to ensure one ID per item?

– How to enforce 1:1 mapping?

– How can it be globally unique?

– How can it be permanent?
Solution I: Central Authority

- GenBank/ EMBL / DDBJ
- May help enforcing 1:1 mapping of an ID and an entity
  - HUGO Nomenclator Committee
  - “Givining unique and meaningful names to every human gene”
- May be infeasible either technically or socially
Solution II: nulD

- Unique, guaranteed
  - Each name identifies only one entity
  - Inherently enforces 1:1 mapping
  - Uniquely resolvable

- Globally unique, guaranteed
  - Decentralized
  - No ID registry necessary

- Permanent, guaranteed

- Carries information about the entity
  - White box
  - no need for a lookup table
How does nuID work?
nuID: the idea

• Sequence itself as the ID
• Combined with the following four features
  – Compression: make it shorter
  – Checksum
    • Prevent transmission error
    • Provide self-identification
  – Encryption: in cases where the sequence identity is proprietary
  – Digital watermark: identify issuer
How does nuID work?

Figure 2
The encoding and decoding process of nuID. The solid arrows represent the encoding process, and the dashed arrows represent the decoding process. The bold-italic number 11 is the numeric value of the checking code "L". The "AA" at the end of sequence is the padded nucleotides.
Example of nuID

<table>
<thead>
<tr>
<th>Array Type</th>
<th>Manufacturer’s Proprietary Identifier</th>
<th>Nucleotide Sequence</th>
<th>nuID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Affymetrix Human</td>
<td>206064_s_at_probe1</td>
<td>TGTATATGTCTGGTTTTCTTACCC</td>
<td>a7M7ev98VQ</td>
</tr>
<tr>
<td>Illumina Human</td>
<td>GI_23097300-A</td>
<td>GCTTCACTCGCTTCCCAGGGGCTCCGTTCACCACACTACATGAGCTACACG</td>
<td>cn0dn1Sqdb0UHE4nEY</td>
</tr>
<tr>
<td>Illumina Mouse</td>
<td>TRBV23_AE000664_T_cell_receptor_beta_variable_23_106-S</td>
<td>GACCCCTTCTGAATGAAAGACACAGTCATGTTATATGGTATAGTCATGGT</td>
<td>9hX2C4CBEtO8zrMtoS</td>
</tr>
</tbody>
</table>
Performance of checksum

Table 2: The error detection power of the nulD checksum algorithm (N = 21)

<table>
<thead>
<tr>
<th>L</th>
<th>1-character</th>
<th>2-character</th>
<th>3-character</th>
<th>Random</th>
</tr>
</thead>
<tbody>
<tr>
<td>25mer</td>
<td>0.97780</td>
<td>0.97918</td>
<td>0.98689</td>
<td>0.99924</td>
</tr>
<tr>
<td>50mer</td>
<td>0.97724</td>
<td>0.97838</td>
<td>0.98607</td>
<td>0.99997</td>
</tr>
<tr>
<td>100mer</td>
<td>0.97894</td>
<td>0.97825</td>
<td>0.98617</td>
<td>1*</td>
</tr>
</tbody>
</table>

*L and N are defined in Equation (3) and (4) in Methods. The column "1-character" is the error detection rate of an nulD with only one character mutated. Similar definition for column "2-character" and "3-character". "Random" column is error detection rate of a random ASCII string. The optimum detection power is 1.0.

*We realize the detection of nulDs for 100mers is not guaranteed. but in none of our simulations did we ever encounter a randomly assembled string that was a valid nulD.
Implementation of nuID

• We have build nuID based annotation packages for all Illumina expression chips.
• We have set up a website for nuID conversion and check latest annotation for the probe.
• The implementation is also included in the lumi package.
Conclusions

• For microarray reporting: Probe-level data is preferred over gene-level data.

• nuID is universal, globally unique, and permanent.

• Do not need a central authority to issue nuID.